
Pengenalan Sains Komputasi - RandomWalk

Ridlo W. Wibowo || 20912009

November 22, 2012

Problem 1.
A hiker without a compass trying to find the way in the dark can step in any of eight directions (N, NE, E, SE, E, SW, W, NW) each step. Studies show that people tend to veer to the right under such circumstances. Initially, the hiker is facing north. Suppose at each step probabilities of going in the indicated directions are as follows: N–19%, NE–24%, E–17%, SE–10%, S–2%, SW–3%, W–10%, NW–15%. Develop a simulation to trace a path of hiker, and run the simulation a number of times. Describe the results. (Note that other than at the initial step, this simulation simplifies the problem by ignoring the direction in which the hiker faces.)

Problem 1.
Perform a simulation of Brownian Motion of a pollen grain suspended in a liquid by generating a three dimensional random walk. Using documentation for your computational tool, investigate how to plot three dimensional graphics and lines, and create a 3D graphic of the walk.

Documentation.
A hiker without a compass
Permasalahan ini menganggap bahwa probabilitas untuk *hiker* melangkah ditentukan oleh probabilitas di atas. Probabilitas tersebut hanya bergantung kepada arah angin (penyederhanaan) tidak berdasarkan kemana *hiker* menghadap (lebih alami). Untuk itu dibuat simulasi *randomwalk* dimana langkah *hiker* dianggap tetap kemanapun ia berjalan dan probabilitas ke arah mana ditentukan dengan cara *roullete* menggunakan distribusi kumulatif sesuai persoalan yang diberikan.

Tiap pengambilan langkah dilakukan dengan melihat angka random yang dicocokkan dengan *roulette* tersebut.

Langkah dan arah tersebut ditentukan dengan menggunakan sudut (koordinat polar), sehingga perlu konversi ke koordinat kartesian setiap waktu, walaupun sebenarnya dalam kasus sederhana ini langkah yang diambil pasti hanya berupa konstanta (langkah *hiker* dianggap tetap). Hal ini membuat program ini tidak terlalu efektif, namun dengan cara ini lebih mudah untuk membayangkan dan memahami permasalahan.

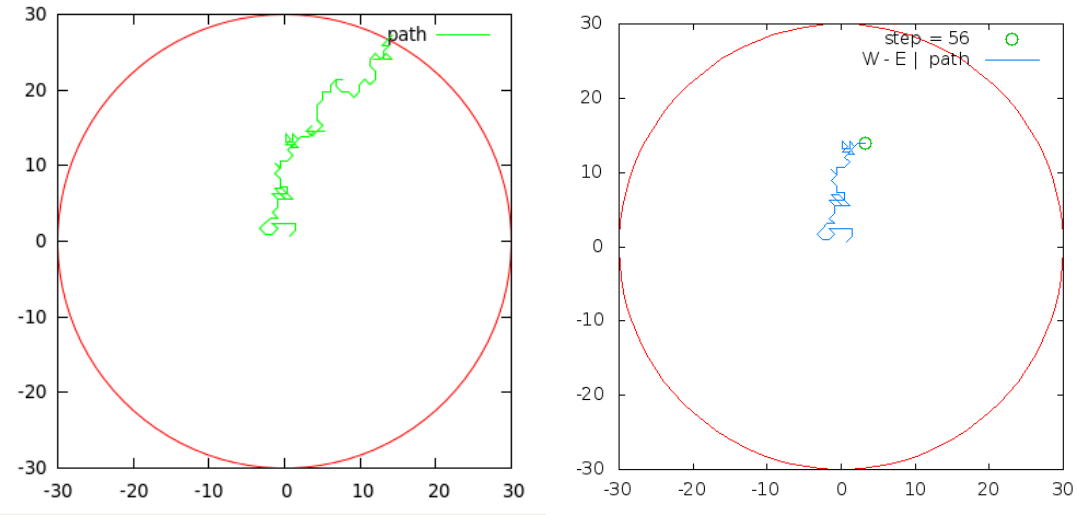
Program dalam bahasa C++ diberikan di bawah ini (berserta fungsi plot dan animasi menggunakan *gnuplot*):

```
/*****************************************/
/* hiker.cpp                             */
/* Copyleft (c) 2012. Ridlo W. Wibowo    */
/*****************************************/
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <fstream>
#define _USE_MATH_DEFINES
using namespace std;

double unirand(){ return (double)rand()/(double)RAND_MAX;}

/*********************************************************/
/* N-19%, NE-24%, E-17%, SE-10%, S-2%, SW-3%, W-10%, NW-15% */
/* cumulative                                            */
double randtheta(){
        double p = unirand();
        if (p < 0.19){ return 0.5*M_PI;} // N
        else if (p < 0.43){ return 0.25*M_PI;} // NE
        else if (p < 0.60){ return 0.0;} // E
        else if (p < 0.70){ return 1.75*M_PI;} // SE
        else if (p < 0.72){ return 1.5*M_PI;} // S
        else if (p < 0.75){ return 1.25*M_PI;} // SW
        else if (p < 0.85){ return M_PI;} // W
        else{ return 0.75*M_PI;} // NW
}

void plot(double Rmax);
void animate(int n, double Rmax);

int main(){
    double step=1.0; // hiker's step is constant: 1 m
    double theta=0.0;
    double Rmax=30;
    double x=0., y=0., dx=0., dy=0., r=0.; // initial at 0,0
```

```
39      int n = 0;
40
41      ofstream out("hiker-out.txt");
42      srand(time(NULL));
43      do{
44          theta = randtheta();
45          dx = step*cos(theta); // actually this method is not effective,
                because dx and dy are just some constants for this problem
46          dy = step*sin(theta);
47          x = x + dx;
48          y = y + dy;
49          out << x << " " << y << endl;
50          r = sqrt(x*x + y*y);
51          n++;
52      } while (r < Rmax); // stop until reach Rmax
53      out.close();
54
55      plot(Rmax);
56      animate(n, Rmax);
57      return 0;
58  }
59
60  void plot(double Rmax){
61      ofstream ploter("hiker-plot.in");
62      ploter << "set parametric\n";
63      ploter << "set xrange [" << -Rmax << ":" << Rmax << "]\n";
64      ploter << "set yrange [" << -Rmax << ":" << Rmax << "]\n";
65      ploter << "set size square\n";
66      ploter << "plot [0:2*pi] " << Rmax << "*sin(t)," << Rmax << "*cos(t) notitle, \
            "hiker-out.txt\" w l title \"path\"\n";
67      ploter.close();
68
69      system("gnuplot -persist < hiker-plot.in");
70  }
71
72  void animate(int n, double Rmax){
73      ofstream anim("hiker-anim.in");
74      anim << "reset\n";
75      anim << "set term gif animate size 500,500\n";
76      anim << "set output \"hiker-animation.gif\"\n";
77      anim << "set parametric\n";
78      anim << "set size square\n";
79      anim << "i=0\n";
80      anim << "n=" << n << "\n";
81      anim << "set xrange [" << -Rmax << ":" << Rmax << "]\n";
82      anim << "set yrange [" << -Rmax << ":" << Rmax << "]\n";
83      anim << "load \"hiker-animate\"\n";
84      anim.close();
85
86      ofstream anime("hiker-animate");
87      anime << "plot [0:2*pi] " << Rmax << "*sin(t)," << Rmax << "*cos(t) notitle, \"
            hiker-out.txt\" every ::i::i ps 2 pt 6 title sprintf(\"step = %i\",
             i), \"hiker-out.txt\" every ::0::i w l ls 3 title \"W - E |  path\
            "\n";
```

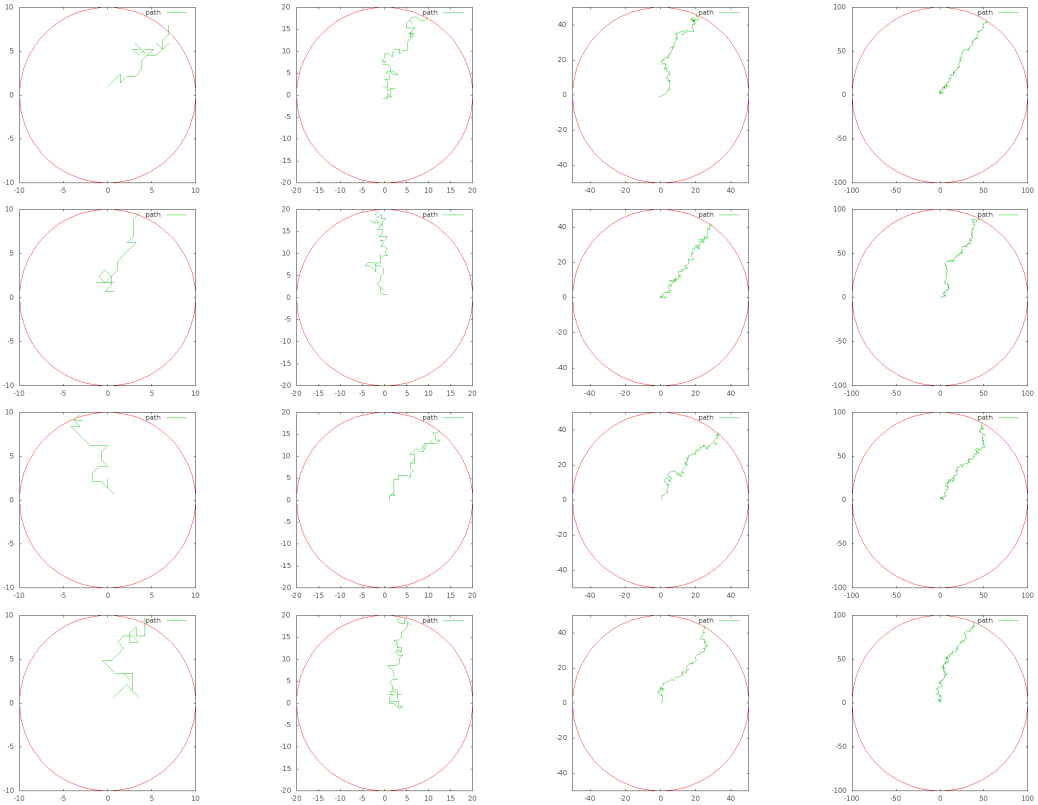
```
88      anime << "i=i+1\n";
89      anime << "if (i < n) reread\n";
90      anime.close();
91
92      system("gnuplot < hiker-anim.in");
93  }
```

Hasil program diatas untuk sekali *run* sebagai berikut:



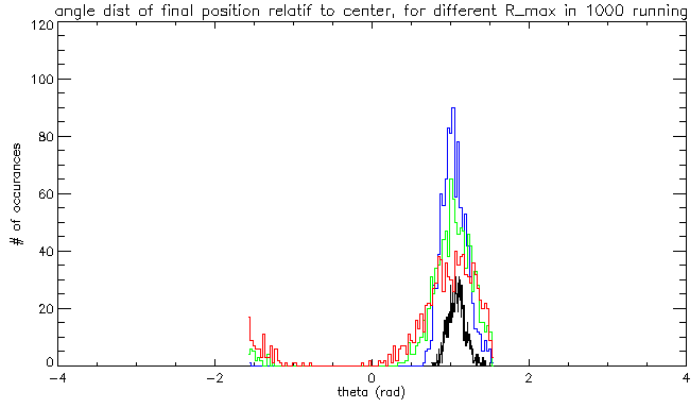
(kiri) Plot untuk batas $R_{max} = 30$, (kanan) cuplikan animasi.

Untuk 1000 kali percobaan dengan R_{max} berbeda,



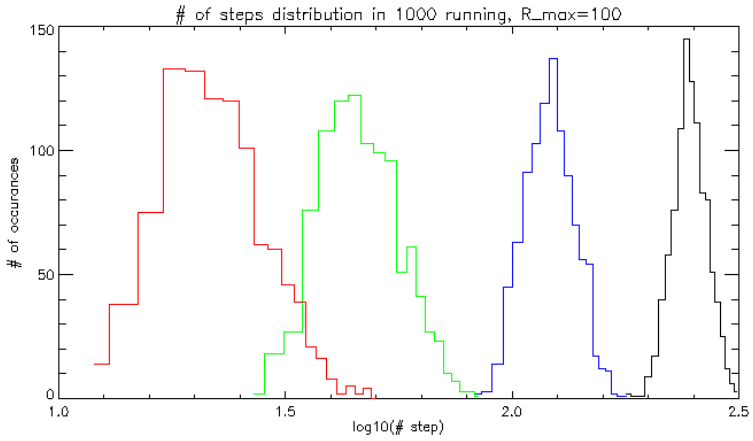
Contoh hasil run dengan R_{max} yang berbeda. Urutan dari paling kiri, $R_{max} = 10, 20, 50, 100$.

Distribusi sudut posisi akhir (di R_{max}) terhadap posisi awal (1000 kali running):



Merah untuk $R_{max} = 10$, Hijau untuk $R_{max} = 20$, Biru untuk $R_{max} = 50$, Hitam untuk $R_{max} = 100$.

Distribusi jumlah step total (1000 kali running):



Merah untuk $R_{max} = 10$, Hijau untuk $R_{max} = 20$, Biru untuk $R_{max} = 50$, Hitam untuk $R_{max} = 100$.

Diskusi

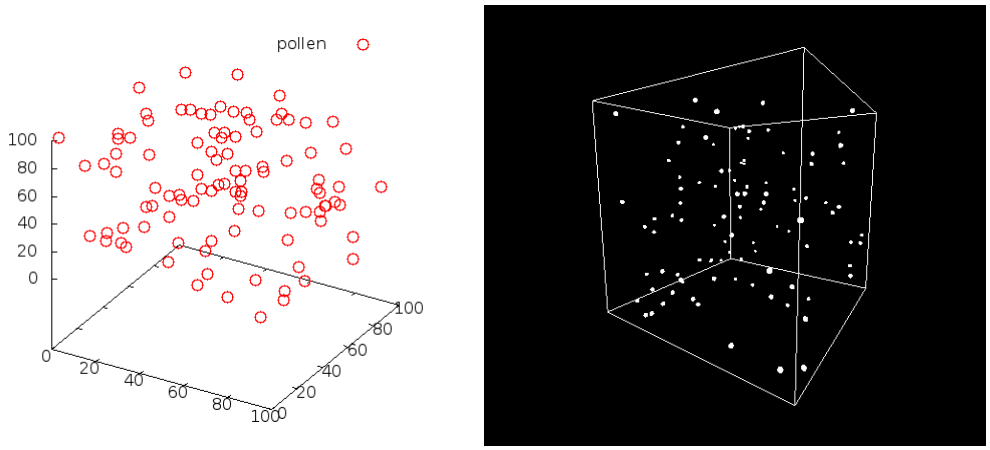
- dari distribusi posisi akhir (arahnya) untuk 1000 kali running diperlihatkan bahwa kecenderungan *hiker* bergerak ke arah sekitar NE.
- semakin lama hiker berjalan ($> R_{max}$) maka distribusinya semakin mengerucut ke arah sekitar NE. dan tidak ada pencilan ke arah sekitar N–NW.
- dari distribusi jumlah step, maka dapat dilihat untuk semakin jauh yang dituju ($> R_{max}$), dibutuhkan jumlah step *randomwalk* lebih banyak, dan terkait secara *power log* (dibutuhkan pemeriksaan lebih lanjut dan fitting).

Brownian motion of pollen in the liquid

Simulasi yang sedikit lebih mendekati kasus sebenarnya dapat dilakukan dengan cara membuat sejumlah partikel kecil sebagai molekul liquidnya dan partikel lain yang jauh lebih besar sebagai pollen. Kemudian dapat dibuat semuanya bergerak dan setiap waktu dicek interaksi tumbukan dari semua pertikel, dan setiap pertikel bertumbukan maka arah gerak dan kecepatannya berubah sesuai momentum awalnya (misal dibuat lenting sempurna). Namun untuk penyederhanaan lagi, kami buat simulasi dengan menggunakan randomwalk 3D saja, namun dengan step yang random pula.

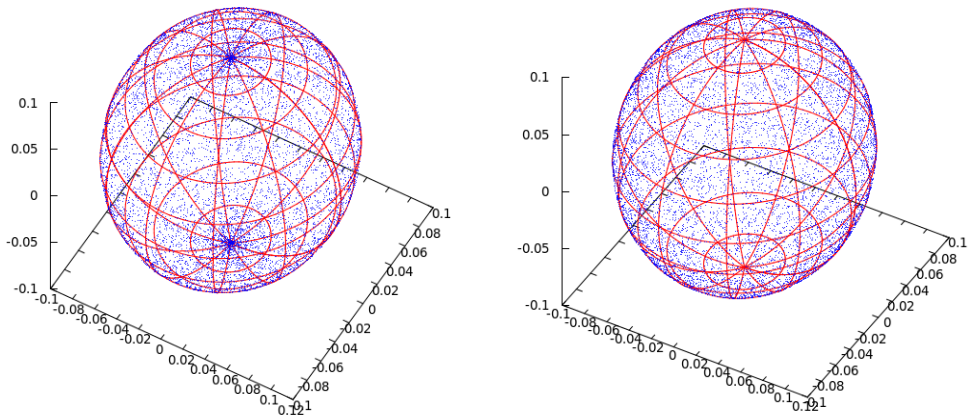
Simulasi randomwalk 100 pollen tanpa interkasi tersebut kami buat dengan C++ dan *gnuplot*, dan juga C++ dengan *OpenGL* (kode terlampir).

Cuplikan hasilnya sebagai berikut:



Simulasi menggunakan *gnuplot* (kiri), dan simulasi menggunakan *OpenGL* (kanan).

Simulasi randomwalk tersebut menggunakan koordinat bola untuk menentukan arah geraknya, sehingga fungsi randomnya perlu dicek terlebih dahulu. Untuk membuat arah random yang benar fungsinya dibuat sebagai berikut:
$\phi = 2\pi r_1$
$\theta = arccos(1 - 2r_2)$
karena apabila θ kita buat hanya berupa random antara 1 dan π maka hasilnya akan tidak random (menghasilkan data di kutub lebih banyak dibandingkan dengan di ekuatornya). Hasil cek yang dilakukan sebagai berikut:



random salah (kiri), random benar (kanan).

LAMPIRAN

Kode untuk gnuplot

```
// Copyleft (c) 2012. Ridlo W. Wibowo
// pollen.cpp
#include<iostream>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<time.h>
#include<fstream>
#define _USE_MATH_DEFINES
using namespace std;

double phi, theta, l;
double unirand(){ return (double)rand()/(double)RAND_MAX;}

int main(){
    int n=100;
    double x[100], y[100], z[100];
    double dx, dy, dz;
    double xm=100., ym=100., zm=100.;
    int t = 0, tf=1000;

    srand(time(NULL));
    // inisiasi
    ofstream out("pollen-out.txt");
    for (int i=0;i<n;i++){
        x[i] = xm*unirand();
        y[i] = ym*unirand();
        z[i] = zm*unirand();
        out << x[i] << " " << y[i] << " " << z[i] << endl;
    }

    do{
  out << "\n\n";
        for (int i=0;i<n;i++){
            phi = 2.*M_PI*unirand();
            theta = acos(1. - 2.*unirand());
            l = unirand();
            dx = l*sin(theta)*cos(phi);
            dy = l*sin(theta)*sin(phi);
            dz = l*cos(theta);

            x[i] = fmod((x[i] + dx + xm),xm); // periodic boundary (toroida
                )
            y[i] = fmod((y[i] + dy + ym),ym);
            z[i] = fmod((z[i] + dz + zm),zm);

            out << x[i] << " " << y[i] << " " << z[i] << endl;
            }

         t = t+1;
    } while(t < tf);
```

```
51      out.close();
52
53      return 0;
54  }
```

Kode untuk OpenGL

```
1  /*************************************************/
2  /* pollen.cpp                                    */
3  /* copyleft (c) 2012. Ridlo W. Wibowo            */
4  /*************************************************/
5  #include<GL/gl.h>
6  #include<GL/glu.h>
7  #include<GL/glut.h>
8  #include<iostream>
9  #include<stdlib.h>
10 #include<math.h>
11 #include<time.h>
12
13 int n=100;
14 double x[100], y[100], z[100];
15 double dx, dy, dz;
16 double l, theta, phi;
17 double unirand(){ return (double)rand()/(double)RAND_MAX;}
18
19 void init(void){
20     glClearColor(0.0, 0.0, 0.0, 0.0);
21     glShadeModel(GL_FLAT);
22 }
23
24 void move(){
25     for (int i=0;i<n;i++){
26         phi = 2.*M_PI*unirand();
27         theta = acos(1. - 2.*unirand());
28         l = 0.05*unirand();
29         dx = l*sin(theta)*cos(phi);
30         dy = l*sin(theta)*sin(phi);
31         dz = l*cos(theta);
32
33         x[i] = fmod((x[i] + dx + 5.),5.); // periodic boundary (toroida)
34         y[i] = fmod((y[i] + dy + 5.),5.);
35         z[i] = fmod((z[i] + dz + 5.),5.);
36     }
37 }
38
39 void myIdleFunc(int a) {
40     move();
41     glutPostRedisplay();
42     glutTimerFunc(100, myIdleFunc, 0);
```

```
43 }
44
45
46 void display(void){
47     glClear(GL_COLOR_BUFFER_BIT);
48     glColor3f (1.0, 1.0, 1.0);
49
50     glutWireCube(5.0);
51     for (int i=0;i<n;i++){
52         glPushMatrix();
53         glTranslatef(-2.5 + x[i], -2.5 + y[i], -2.5 + z[i]);
54         glutSolidSphere(0.05, 10, 8);
55         glPopMatrix();
56     }
57     glutSwapBuffers();
58 }
59
60 void inisiasi(){
61     for (int i=0;i<n;i++){
62         x[i] = 5.*unirand();
63         y[i] = 5.*unirand();
64         z[i] = 5.*unirand();
65     }
66 }
67
68
69 void reshape(int w, int h){
70     glViewport(0, 0, (GLsizei) w, (GLsizei) h);
71     glMatrixMode (GL_PROJECTION);
72     glLoadIdentity();
73     gluPerspective(100.0, (GLfloat) w/ (GLfloat) h, 1.0, 20.0);
74     glMatrixMode(GL_MODELVIEW);
75     glLoadIdentity();
76     gluLookAt(4.0, 1.0, 6.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
77 }
78
79 int main(int argc, char** argv){
80     glutInit(&argc, argv);
81     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
82     glutInitWindowSize (700, 700);
83     glutInitWindowPosition (100, 100);
84     glutCreateWindow ("pollen brownian motion");
85     init ();
86     srand(time(NULL));
87     inisiasi();
88     glutReshapeFunc(reshape);
89     glutDisplayFunc(display);
90     glutTimerFunc(100, myIdleFunc, 0);
91
92     glutMainLoop();
93     return 0;
94 }
```

Simulasi kami upload di `http://astrokode.wordpress.com/`